Politecnico di Torino

Digital systems electronics
A.A. 2018/2019

Prof. G. Masera

Lab 02

26 Mar 2019

Berchialla Luca	236032
Laurasi Gjergji	238259
Mattei Andrea	233755
Lombardo Domenico Maria	233959

1 Controlling a 7-segments display

Figure 1 shows a 7-segment decoder module whose input bits $C2C1C0$ drive a 7 segment display through the bits $HEX0_0 -> HEX0_6$

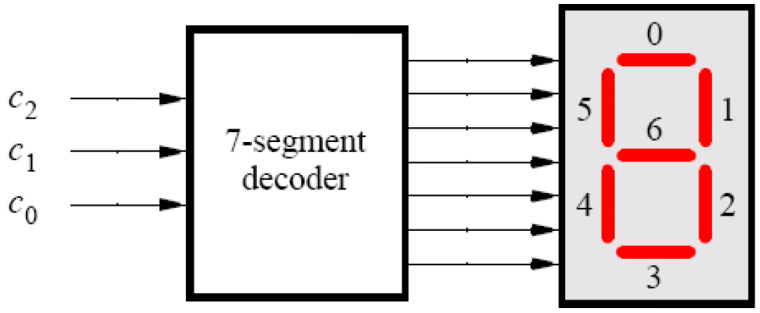

Figure 1: 7-segment decoder + display

Figure 2 shows the truth table to be implemented for the 7-segment decoder. As shown just the characters $HELO$ will be implemented. The sequent logical states can be easily derived from the table:

$$HEX6 = \overline{C2} \cdot \overline{C1}$$
$$HEX5 = \overline{C2}$$
$$HEX4 = \overline{C2}$$
$$HEX3 = \overline{C2} \cdot (C0 + C1)$$
$$HEX2 = \overline{C2} \cdot \overline{C1} \cdot \overline{C0} + \cdot\overline{C2} \cdot C1 \cdot C0$$
$$HEX1 = \overline{C2} \cdot \overline{C1} \cdot \overline{C0} + \cdot\overline{C2} \cdot C1 \cdot C0$$
$$HEX0 = \overline{C2} \cdot C0$$

C2	C1	C0	HEX6	HEX5	HEX4	HEX3	HEX2	HEX1	HEX0
0	0	0	1	1	1	0	1	1	0
0	0	1	1	1	1	1	0	0	1
0	1	0	0	1	1	1	0	0	0
0	1	1	0	1	1	1	1	1	1
1	X	X	0	0	0	0	0	0	0

Figure 2: decoder truth table

Finally the logic states are implemented using gates as shown in $figure3$.
The circuit is then described into VHDL using a dataflow style approach, the VHDL file is called *puntoA.vhd*.

The VHDL entry has been finally simulated via *testbenchapproach* where every possible input combination has been considered validating the output. The testbench results are shown below in figure 4.

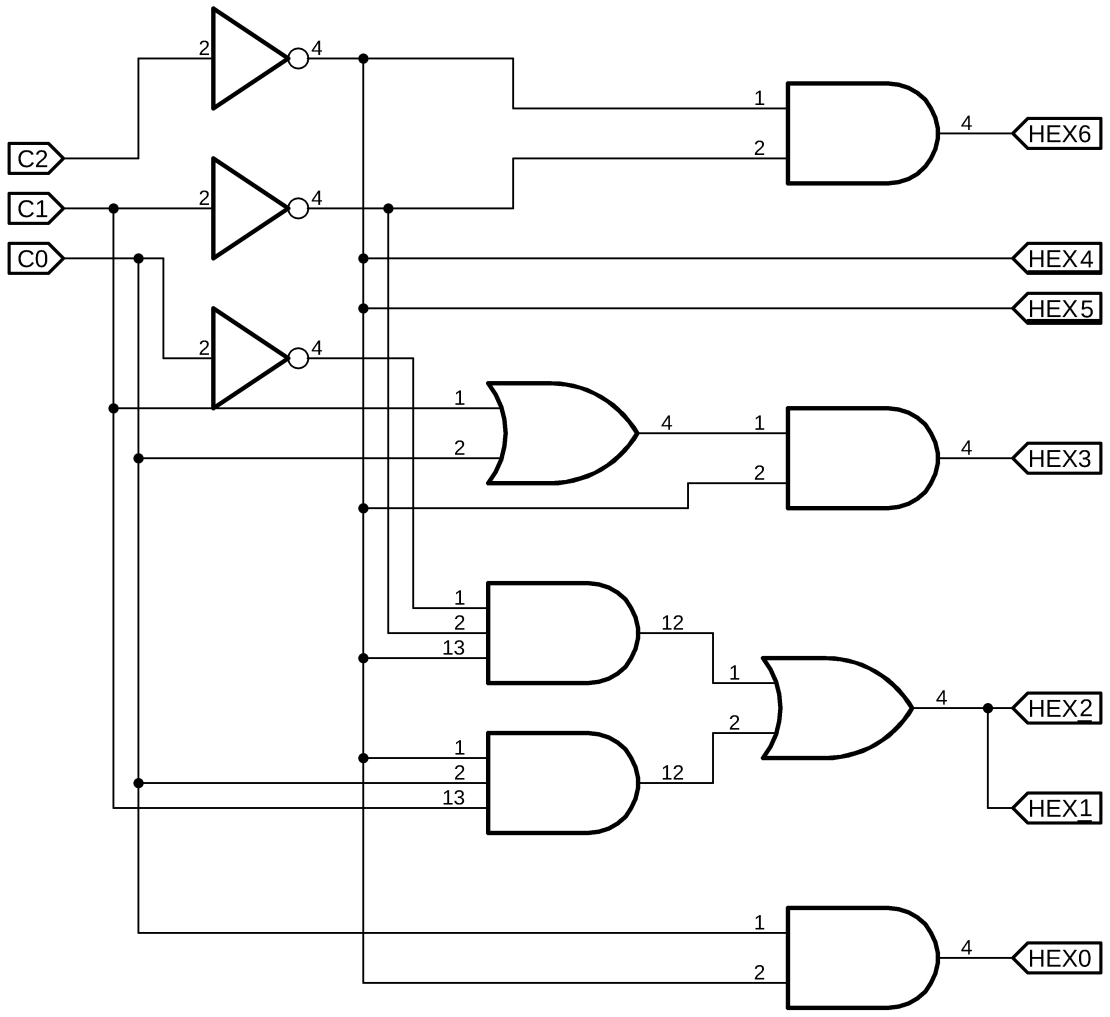

Figure 3: decoder gates implementation

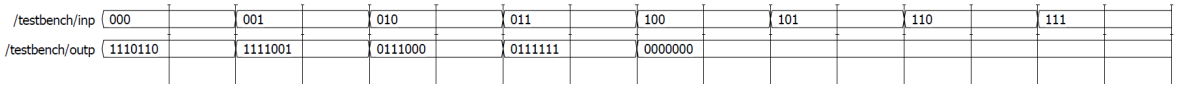

Figure 4: Testbench results

2 Multiplexing the 7-segments display output

Figure 5 shows the architecture of the implemented circuit. It is composed by a three-bit wide 4-to-1 multiplexer, which has the data inputs fixed to the words we want to display following the codification of figure 6. A shifter that rotates the selected word in a circular fashion and 5 7-segments decoders followed by their respective displays.

Each sub-circuit has been implemented and tested separately than in the *Part*2

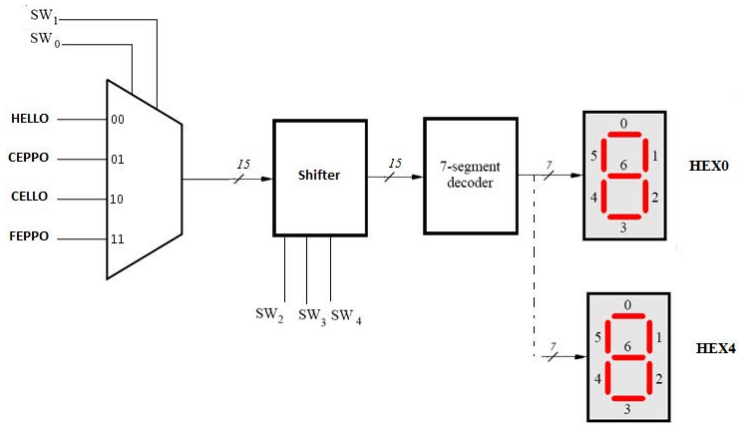

Figure 5: multiplexer + shifter + 7-segment decoder + display

IN	Character
000	H
001	E
010	L
011	O
100	C
101	P
110	F
111	

Figure 6: 7-segment display character codes

entity all components are used to describe the circuit.

2.1 Multiplexer

Each letter code has been assigned to a signal for code readability. Than the output of the multiplexer has been described using the *when* structure.
The testbench of this component assigns all possible combinations of the selection inputs, then the output is checked using the waveforms.

2.2 Shifter

The shifter output has been set as a concatenation of two parts of the input for meaningful input and forced to zero for all other inputs. The behaviour implemented is described in $figure7$. This component has been tested by fixing the input to a concatenation of 5 3-bit binary numbers in rising order and assigning all possible combination of the SW inputs. Than the behaviour has been checked in the simulated waveforms.

SW4 SW3 SW2	Character pattern
000	HELLO
001	ELLOH
010	LLOHE
011	LOHEL
100	OHELL

Figure 7: Example of shifter behaviour with the word "HELLO"

2.3 7-segment decoder

This component could have been reused from the previous point but it has been quickly reimplemented behaviourally to be able to display also the letters C,P, and F. Although, the testbench was reused taking care to verify also the new letters in the waveform simulation.

2.4 Testbench of the whole circuit

To be able to verify the correct behaviour of the circuit several signals have been created.
A signal for each letter with values for each letter corresponding to the 7 bits of the display code. An *error* signal that is used as a flag to quickly detect possible errors. In the port map the displays are assigned in inverse order to test view the output in the simulation with the first letter in the $HEX0$ display instead of the $HEX4$ display.
The circuit first tested by fixing the multiplexer selection to 00 corresponding to the $HELLO$ word and assigning all the possible combinations to the shifter selection inputs. With several if statements the cycling of the $HELLO$ word is checked.
Than the shifter selection inputs are forced to 000 and is verified that all the other words are displayed properly.

3 Binary to Decimal converter

The task of this part consisted in making a combinational circuit (converter.vhd) that converts a 4-bit binary number into its decimal numeral system form and then displays the result using 2 7-segments displays. The circuit is composed of 3 elements: the converter that performs the binary to decimal conversion (inner_converter.vhd), a multiplexer used for driving the display associated to the tens digit (tens_multiplexer.vhd) and a BCD to 7-segments encoding converter used for driving the display associated to the units digit (bcd_to_7s.vhd).
The inner convert takes a 4-bit binary number as input and decomposes it into its tens and units digits. Since the maximum input number is 15, the circuit returns just 1 bit in output for the decimal digit. The output for the units digit is instead

a 4-bit binary number ranged between 0 and 9.
The tens multiplexer drives the 7-segments display. The received 1-bit output of the inner converter selects the corresponding digit, encoded in the 7-segments standard, to display and returns it as output.
Finally, the BCD to 7-segments converter takes the binary encoded units digit and converts it into the 7-segments encoding. The conversion is performed using the "with select"construct of VHDL.
Given the relatively small range of inputs, the testbench used for testing tried every possible input by iterating with a for loop.

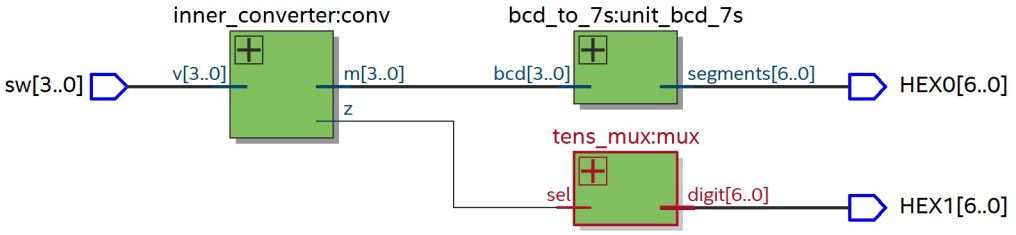

Figure 8: the RTL schematic of the binary to decimal converter generated by Quartus Prime

4 Binary-to-BCD Converter

A 6 bit Binary to decimal decoder has been implemented using the "$ieee_numeric_std\ library$ needed to compute subtraction operations between binary numbers.
The subtraction permits to find the unit digit.

Finally a testbench checks all the possible combination of numbers $0 \rightarrow 63$ by means of a process statement.